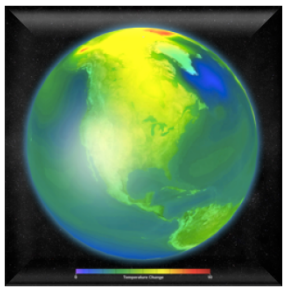
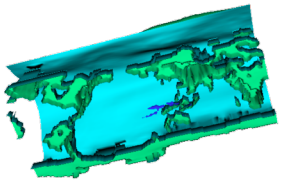
Lawrence Livermore National Laboratory
LLNL-AR-466371

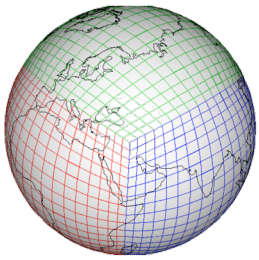
Ultra-scale Visualization Climate Data Analysis Tools (UV-CDAT)

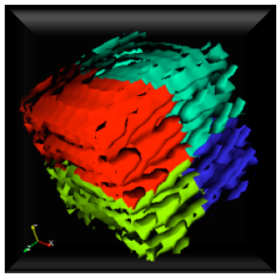
Semi-Annual Progress Report

July 1, 2010 through December 31, 2010









UVCDAT

LLNL · Oak Ridge · Los Alamos · University of Utah · NYU-poly · VT · ParaView · Kitware

BER's Ultra-scale Visualization Climate Data Analysis Tools

Semi-Annual Progress Report for the Period

July 1, 2010 through December 31, 2010

Principal Investigators

Dean N. Williams[3], David Bader[5], and Phil Jones[2]

The Ultra-scale Visualization Climate Data Analysis Tools Team:

Andy Bauer[1], Berk Geveci[1], Dave Partka[1]
Jim Ahrens[2], John Patchett, Johnathan Woodring[2]
Timo Bremer[3], Charles Doutriaux[3], Robert Drach[3]
Thomas Maxwell[4]
Ross Miller[5], Galen Shipman[5], Raju Ranga Vatsavai[5], Feiyi Wang[5], Zhe Zhang[5]
Cameron Christensen[6], David Koop[6], Valerio Pascucci[6], Emanuele Santos[6],
Claudio Silva[6], Huy Vo[6]

[1] Kitware, Inc.
[2] Los Alamos National Laboratory
[3] Lawrence Livermore National Laboratory
[4] National Aeronautics and Space Administration
[5] Oak Ridge National Laboratory
[6] University of Utah

Table of Contents

1 Executive Summary

This report summarizes work carried out by the Ultra-scale Visualization Climate Data Analysis Tools (UV-CDAT) for the period July 1, 2010 through December 31, 2010. It discusses highlights, overall progress, period goals, and collaborations, and lists papers and presentations. To learn more about our project, please visit our UV-CDAT websites (URL: http://uv-cdat.org). This report will be forwarded to the program manager for the Department of Energy (DOE) Office of Biological and Environmental Research (BER), national and international collaborators and stakeholders, and to researchers working on a wide range of other climate model, reanalysis, and observation evaluation activities.

The UV-CDAT executive committee consists of Dean N. Williams, Lawrence Livermore National Laboratory (LLNL); Dave Bader, Oak Ridge National Laboratory (ORNL); and Phil Jones, Los Alamos National Laboratory (LANL). The UV-CDAT team is a group of researchers and scientists with diverse domain knowledge, whose home institutions include four laboratories, a university, and a private company: LANL, LLNL, National Aeronautics and Space Administration (NASA), ORNL, University of Utah, and Kitware Inc. All work is accomplished under the DOE open-source guidelines and in close collaboration with the project's stakeholders, domain researchers, and scientists.

Working directly with BER climate science analysis projects, this consortium will develop and deploy data and computational resources useful to a wide variety of stakeholders, including scientists, policymakers, and the general public. Members of this consortium already collaborate with other institutions and universities in researching data discovery, management, visualization, workflow analysis, and provenance. The UV-CDAT team will address the following high-level visualization requirements:

- Alternative statistics and analysis methods
- Data intercomparison visualization
- Optimized parallel input/output (I/O)
- Parallel visualization
- Data provenance processing
- Workflow data analysis and visualization
- Advanced parallel scientific visualization

2 Reporting Period Highlights from Kitware, LANL, LLNL, NASA, ORNL, and the University of Utah

This section describes the team's accomplishments during the reporting period, where the overall goal was to move the project forward to accomplish its mission of providing useful tools and product services within the UV-CDAT framework.

The UV-CDAT is intended to serve customers who span a broad spectrum of sophistication. These users range from numerical modelers who want access and analysis to "raw" model output files and verbatim subsets of model output, to climate impacts investigators who want rapid access to these data and analysis without the complexities of

model-specific coordinate systems, and users who only want to quickly visualize the overall behaviors of model and observation intercomparisons. The ultrascale nature of climate data holdings requires that significant levels of data reduction to take place at the server so we can satisfy these customers—both through straightforward subsetting and decimation and through specific analysis operations, such as the computing of spatio-temporal averages. In the UV-CDAT architecture, we refer to the steps that convert raw data into analysis results and visualizations as "product services." Detailed use cases, architecture design, and software challenges can be found on our UV-CDAT website: http://uv-cdat.org.

2.1 Kitware and LANL project highlights include:

- Update of VTK's NetCDFPOP reader – improved efficiency and fixed some bugs
- Initial design of temporal parallelism – started the design of temporal parallelism to improve performance on leadership class supercomputers
- ParaView/VisTrails integration – laying the groundwork
- Integration of software build and testing tools
- UV-CDAT team collaboration involvement
    - Parallel planning
    - Planning and support of new visualization methods
    - Planning and support of parallel I/O
    - Provided data for use-cases 1 and 2
    - Developed use-case documentation

2.2 LLNL project highlights include:

- Worked with LANL and others to create detailed specific use cases needed for high-level UV-CDAT system functions and project requirements. Use cases include:
    - Compute ensemble mean
    - Compute a time series of a regional average
    - Compute a zonal mean
    - Convert from hybrid to standard pressure levels
    - Compute departures from climatological boreal winter
    - Compute average multi-model ensemble mean
    - High spatial resolution, parallel, time average
    - High spatial resolution, parallel, image sequence production
- Rewriting the VCDAT graphical users interface (GUI) and event manager interface to work within the VisTrail environment. (Replacing the X11R6 windowing interface with Qt.) Initial start completed. Enough done to release an alpha version of UV-CDAT in a few months.
- Working with Kitware to develop a robust and stable UV-CDAT code builder process via CMAKE.

- Generated the beginnings of the UV-CDAT website—to contain essential documents relevant to UV-CDAT and coordinating activities. The website can be found at: http://uv-cdat.org.
    - A wiki has been set up to allow group access and editing of important documents, such as use cases and sub-projects (CDAT, VisTrails, ParaView, R, VisIt, etc.)
- Created a GIT code repository for each UV-CDAT core software component. The core component code is open-source and distributed via the "git" repository system: for example, for the CDAT software component, the URL is:
    - git checkout git@uv-cdat.llnl.gov:uv-cdat.git
- Set up "bugzilla" to allow users and developers to report UV-CDAT bugs and enhancements and allow developers to easily identify and track code corrections.

2.3 NASA project highlights include:

- Developing high-level modules for UV-CDAT to provide user-friendly interfaces for 3D visualization. This work includes:
    - Drag slice planes along coordinate axes
    - Resize and rotate slice planes using key + mouse
    - Adjust the scaling of color maps using key + mouse drag
    - Adjust the volume render transfer function using key + mouse drag
    - Access custom VisTrails module GUIs for choosing color map, etc.
    - Greatly simplified (with regards to VTK) workflows

2.4 ORNL project highlights include:

- Completed proof of concept prototype of temporal parallelism using ParaView on the Jaguar XT5 leadership computing platform.
    - Demonstrated near-linear speedup (up to 36x better performance) of climate data analysis workload using this prototype
    - Optimized Parallel I/O for large-scale temporal parallelism
    - Collaborating with LANL and Kitware on analyzing these results and implications for the temporal parallel architecture development within ParaView/VTK
- Developing optimized Parallel I/O infrastructure in ParaView/VTK
    - Completed initial analysis of Parallel I/O performance in ParaView/VTK using NetCDF
    - Developed prototype implementation of PnetCDF (Parallel NetCDF) reader in ParaView/VTK
    - Created standalone Parallel I/O kernel for VTK's NetCDFPOP reader
        - Evaluating a number of I/O access and scheduling mechanisms
        - Evaluating alternative data decomposition techniques

- Collaborating with LANL and Kitware on analyzing results and determining alternative data decompositions

- Facilitated access to Jaguar XT5 and other ORNL systems for collaborative development through a Director's discretionary allocation (PI: Galen Shipman)

2.5 The UV-CDAT team is now utilizing the computational and storage resources at the Oak Ridge Leadership Computing Facility to support our efforts.University of Utah and LLNL project highlights include:

- Collaborated with Dean N. Williams (LLNL) on the UV-CDAT logo. SCI Institute graphics designer Nathan Galli did the artwork.
- Participated in the architecture meeting and helped to prototype an integrated UV-CDAT tool that includes CDAT, ParaView, and VisTrails.
- Together with Dean Williams and Charles Doutriax (LLNL), designing a scripting interface for creating CDAT workflows that enables easier provenance capture.
- Together with Berk Geveci (Kitware), created a VisTrails package for ParaView, including Spreadsheet integration. This package enables UV-CDAT to seamlessly use ParaView large-scale rendering functionality.

3 Component Progress

During this reporting period, progress was made in the key areas that are necessary to meet UV-CDAT goals and objectives.

3.1 VTK NetCDF POP Reader

The NetCDF POP reader in VTK (vtkNetCDFPOPReader) was reviewed for correctness. After review it was determined that the reader was able to work in parallel and with strided reading (i.e., reading a regularized subset of the data file). It did not work properly, though, with striding in parallel, and this problem was fixed. In addition, memory leaks were eliminated and the code was refactored to improve performance and code maintainability. In addition, a test was added for the reader to ensure that future changes will not introduce bugs. We are currently verifying that the reader is properly interpreting the file contents.

3.2 NetCDF Climate and Forecast (CF) convention compliance

UV-CDAT will be based on NetCDF CF compliance. The components and tools will "assume" compliance in order to make assumptions about the data. For example, ParaView I/O needs to be aware of CF metadata in order to make important aggregations and analysis prior to visualization. Full NetCDF CF compliance work will include Gridspec—a standard description of grids (e.g., Rectilinear, Gaussian, Tripolar, Icosahedral, and Cubed Sphere) used in Earth System models.

3.3 Parallel input and output (I/O)

The team has been working closely to define the overall UV-CDAT architecture, in particular the tightly coupled component framework. Early work has identified two key optimizations for ultra-scale climate analysis that the ORNL and LANL team have focused on. The first optimization utilizes temporal parallelization of climate data analysis workloads. To demonstrate the benefits on temporal parallelization series, we are running multiple ParaView servers in parallel, where each ParaView instance is assigned a distinct temporal domain. As POP decomposes the temporal output into distinct POP NetCDF files, each ParaView instance only accessed a single POP NetCDF file. Our experiments showed near-linear speedup for up to 36 simultaneous analysis tasks across the temporal dimension. The second optimization that the ORNL team has explored is aimed at improving the I/O performance of spatial parallel climate analysis workloads.

Our initial analysis of a ParaView analysis workload of a POP simulation on Jaguar XT5 at ORNL indicates that the majority of time is spent on I/O reading of a POP NetCDF file from multiple processes. More than 74% of the total runtime is attributable to reading the POP NetCDF file, with the majority of the time spent in the nc_get_vara_float procedure. The ORNL team is currently working with the LANL and Kitware teams to explore alternative domain decompositions to improve I/O performance. A prototype I/O kernel has been developed that allows the evaluation of alternate domain decompositions and their impact on overall I/O performance. This kernel will allow rapid prototyping and will then inform development of alternative domain decompositions (if required) within the ParaView system.

3.4 Temporal Parallelism Design

There have been two major mechanisms for introducing parallelism to data analysis workflows:

- Data parallelism, where the spatial grid for each time step is partitioned across processes
- Temporal parallelism, where each process is assigned the whole grid for different time steps

Computationally, the two parallelisms are very similar as long as load balancing is done carefully. However, when I/O and inter-process communication are considered, there may be significant differences between the two approaches. For example, when processing a time series of small- to medium-size data sets in which each time step is written as one NetCDF file, it is more efficient to utilize temporal parallelism. Assigning separate files to different processes allows I/O to scale better than when all processes are reading each file in parallel. On the other hand, when processing a time series of medium- to large-size data sets, it may not even be possible to load one time step in memory, and data parallelism is required.

ParaView has successfully demonstrated the use of data parallelism to achieve scalability, but it does not currently support temporal parallelism. We have therefore started investigating the use of temporal and hybrid parallelism within ParaView in order to obtain maximum performance on the leading edge supercomputing machines.

3.5 Provenance

The goal of this task is to generate both the prospective and retrospective provenance of current UV-CDAT Python scripts in VisTrails. This task is composed of two subtasks: encapsulating UV-CDAT functionality in VisTrails packages and translating UV-CDAT Python scripts to the structured workflow format.

A VisTrails package is similar to a Python package, which defines a set of classes and functions (VisTrails modules) for a particular interface. These modules are used by VisTrails to build workflows using UV-CDAT functionality.

We are building the UV-CDAT VisTrails packages automatically based on Python's introspection and/or XML descriptions provided by CDAT Python modules. These descriptions detail their input and output types, as well as required and optional inputs. A parser to build corresponding VisTrails modules uses this information. Work has been done on two fronts:

- Enabling all functions in UV-CDAT where introspection is not available to generate xml descriptions
- Making sure they are correctly parsed and the UV-CDAT packages are functional in VisTrails

Migrating scripts to a provenance-enabled workflow format requires a two-step translation, as shown in the figure below. The first step involves translating from original Python script to VisTrails Shell script, an intermediate format that will allow direct translation to the structured workflow format. The VCDAT window will generate Python commands (in original Python script format) according to the GUI actions performed by users, who will also use this mechanism directly to generate provenance.

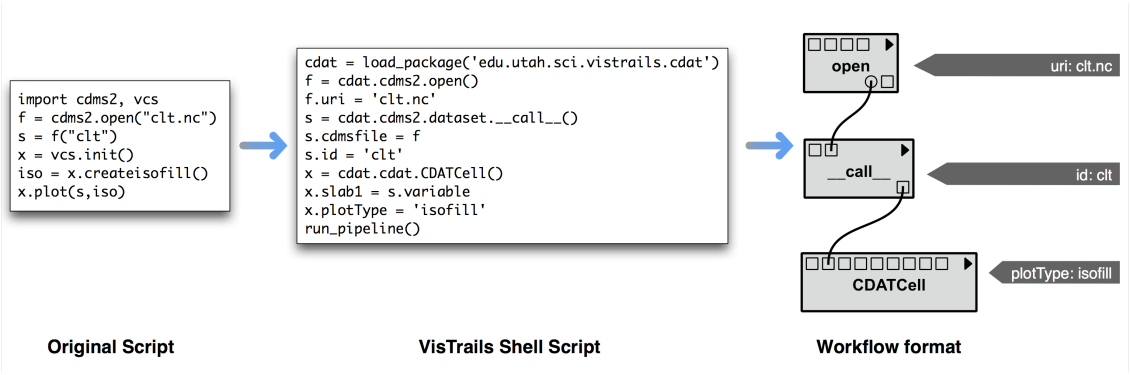


Figure 1: *Simple UV-CDAT Python script opening a file and plotting a two-dimensional contour plot*

Bare-bones communication between CDAT and VisTrails is in place. Most of CDAT's doc strings still need to be updated to allow better communication with VisTrails. A wiki page has been set up to describe the appropriate way for UV-CDAT components to communicate with VisTrails. (See the URL: http://uv-cdat.llnl.gov/wiki/VistrailsXMLDescriptions for details.)

3.6 UV-CDAT graphical user interface

The rewriting of the graphical user interface and event manager interface is underway. Rather than completely taking out X11 and replacing it with Qt, a more "modular" approach has been chosen: simply using pre-processor calls to direct the appropriate back-end component. At the moment, this back end can be either X11 or Qt, with the door open for future technology "upgrades" or innovations.

As far as VCDAT is concerned, a rewrite from the ground up has been determined as necessary; this rewrite allows for a cleanup of both the user interface and VCDAT's inner components. While advanced features of the old VCDAT are not yet in place, many basic features have already been re-implemented. See Figure 2 for comparison.

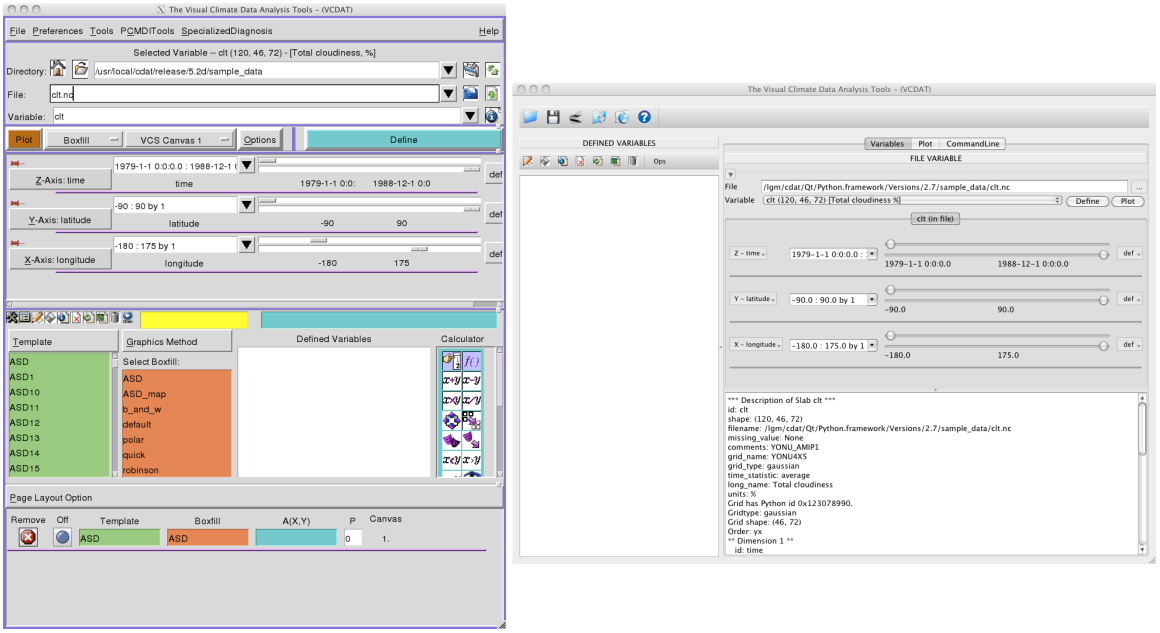


Figure 2: *Old vs. new VCDAT*

3.7 Integration of Software Build and Testing Tools

Development of robust and stable code at a large scale requires a sound software process. All the groups developing the major architecture components of UV-CDAT have their own software process that will be unified as part of this effort. We are in the process of adopting and extending the Kitware software process model.

To develop large libraries, Kitware uses a novel process based on methods from agile development and test-driven development. The process incorporates source code control (Subversion (SVN) or Git, for example), build management using CMake, and regression testing using CTest and CDash. One of the highlights of this process is that it runs continuously and simultaneously on multiple platforms and produces its results on web-accessible dashboards. Thus, as software is checked into the source code repository, it is immediately tested and results posted. Developers rapidly obtain feedback on their changes and correct problems immediately. This process is demonstrated in Figure 3.

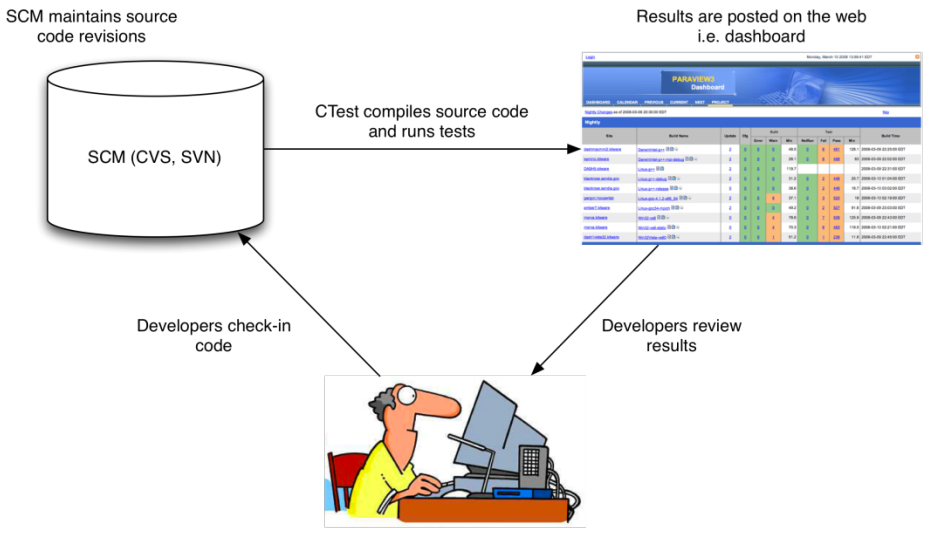


Figure 3: *Test-driven agile software process*

The following are the important components in this process:

- CMake is a family of open-source tools designed to build, test, and package software. CMake is used to control the software compilation process using simple platform and compiler-independent configuration files. CMake generates native makefiles and workspaces that can be used in the compiler environment of the developer's choice. CMake is quite sophisticated: it is possible to support complex environments requiring system introspection, pre-processor generation, code generation, template instantiation, and cross-compilation.
- CTest is a testing tool distributed as a part of CMake. It can be used to automate updating (using SVN, for example), configuring, building, testing, performing memory checking, performing coverage, and submitting results to the CDash dashboard system.
- CDash is an open source, web-based software testing server. CDash aggregates, analyzes, and displays the results of software testing processes submitted from clients located around the world.

As a first step towards introducing an agile software development process to UV-CDAT, we are working on converting its build process to use CMake. This conversion will allow us to use CTest and CDash in the future.

For UV-CDAT, we will employ CMake 2.8's ExternalProject module to build each of its approximately 40 dependencies. The ExternalProject module makes it easier to build external software components on which the project depends. An "external project" is one for which the source code is available but is not necessarily part of the main build process. The ExternalProject_Add function makes it possible to say, "Download this project from the Internet, run its configure step, build it, and install it," in just a few lines of code in the CMakeLists.txt file. All of the time-intensive processing that occurs for each step is deferred until build time.

The basic concept of ExternalProject is simple: given an external source of software (URL to a .tar.gz file, Concurrent Version Control (CVS) repository, SVN repository, or local

directory), execute the sequence of commands necessary to build and install that software such that it can be referred to (include, link, run) from the main project.

By design, ExternalProject is modular and compact. With it, we can easily extend UV-CDAT to build additional packages when the need arises. Additionally, just employing CMake as the build system of UV-CDAT allows us to very easily extend and customize its build process in a variety of ways. We can use CMake to give the users numerous build options, allowing a developer to easily build portions of UV-CDAT as needed. This will allow us to achieve the goal of distributing a "UV-CDAT-lite."

The current status is that we have written a CMake Version of UV-CDAT's build system. This build system can download, configure, and build UV-CDAT along with all of its 40 dependencies. The new build system will allow us to add options to easily customize UV-CDAT as well as extend it with new dependencies.

3.8 ParaView and VisTrails Integration

The UV-CDAT framework supports both loosely coupled and tightly coupled dataflow fragments. The loosely coupled dataflow is built on top of the VisTrails dataflow model, while the tightly coupled data flows are built on top of the VTK/ParaView dataflow network. Tightly coupled integration of the CDAT Core with the VTK/ParaView infrastructure provides high-performance parallel streaming data analysis and visualization of massive climate data sets. Within both paradigms, UV-CDAT will provide data provenance capture and mechanisms to support data analysis via the VisTrails infrastructure.

The VisTrails/ParaView integration in UV-CDAT is done by loosely coupling ParaView with other packages within the VisTrails dataflow model. We have built an initial version of a ParaView package that enables provenance capture and interoperability of ParaView with other UV-CDAT packages. The current version of the ParaView package has been integrated into the Spreadsheet package. It works as a normal ParaView client, connecting with the ParaView server for all its computational needs. All ParaView server resources are accessible by UV-CDAT.

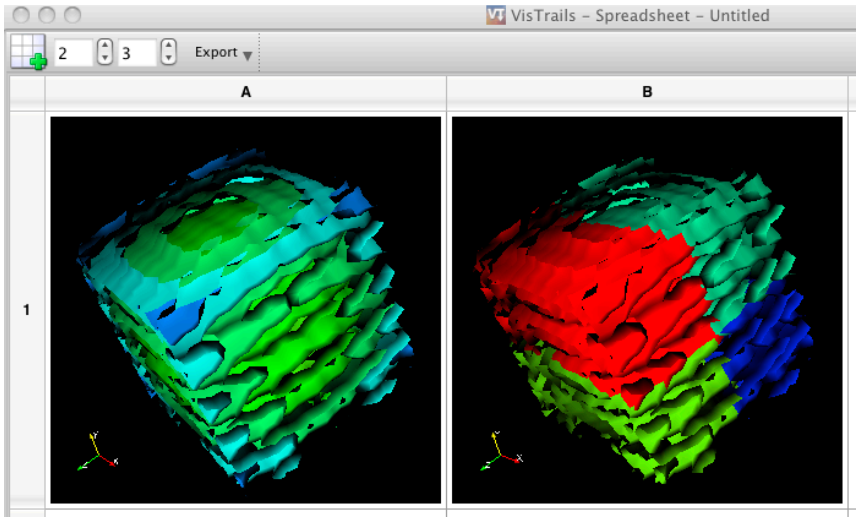
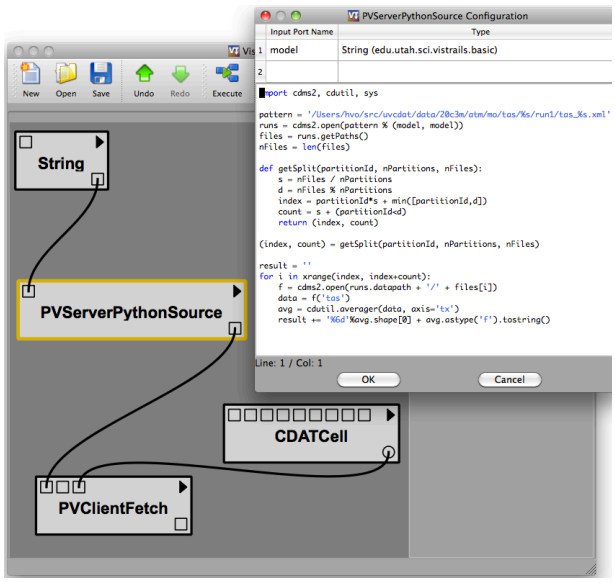
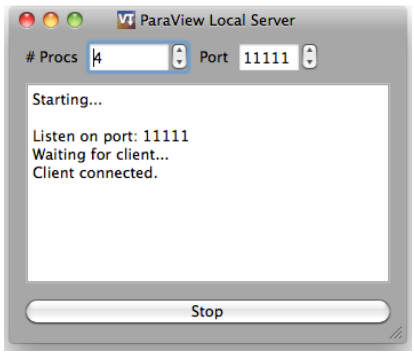


Figure 4: *ParaView in VisTrails*

Figure 4 demonstrates the integration of ParaView with VisTrails. Here the ParaView module is connected to a four-process ParaView server running on a remote cluster.

In the current version, we can use ParaView's ProgrammableFilter to send arbitrary Python code to the ParaView server in order to parallelize CDAT functionality. We have also created a simple GUI to control the number of server processes running. The computation runs on the server processes, but the visualization is done on the client side. Since ProgrammableFilter does not have any configurable I/O ports, we create a PVServerPythonSource module to support this on top of the ProgrammableFilter. We also added a quick way to convert arbitrary Python strings to/from vtkImageData for use with these ProgrammableFilters. The results of these filters can later be collected and further processed on the client side through our PVClientFetch module.





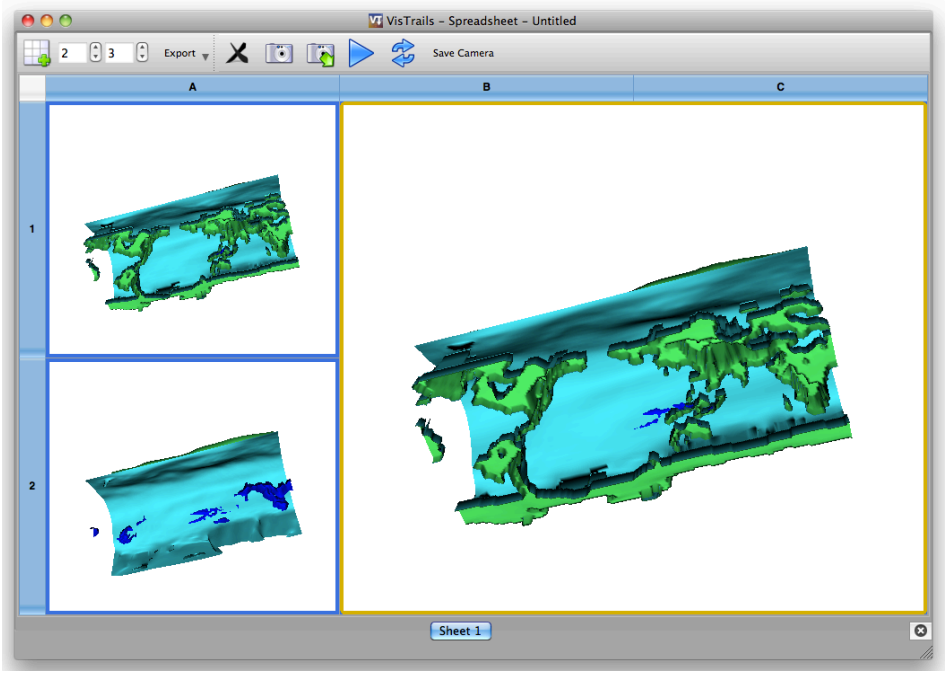


Figure 5: *ParaView pipeline connections in VisTrails*

3.9 Website(s)

The main website needs verbiage about the project; we will probably just copy and paste from the proposal(s). Since the website is shared by both proposals, we need to decide how to clearly mark what is UV-CDAT and what is Data Explorer. For the most part the distinction is drawn as follows: UV-CDAT components include CDAT, VisTrails, and ParaView and Data Explorer components include the addition of VisIt and R.

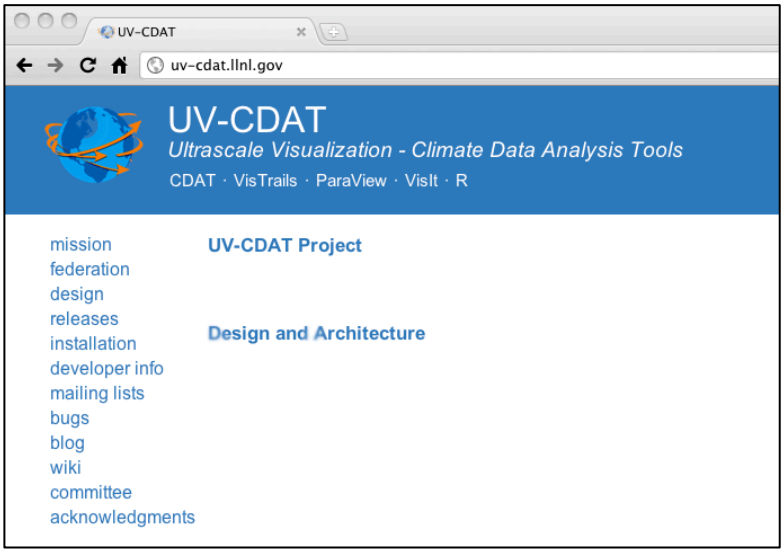


Figure 6: *UV-CDAT website*

Each component's web section is built around the same template and is stored under a Git (or "git") repository. Each component needs to designate a main "webmaster" that will upload the initial content and maintain the overall look afterward. The procedure to check out the repository for example for the CDAT web:

- git checkout git@uv-cdat.llnl.gov:[component]-site.git

At the moment, there is the top [UV-CDAT] component and five sub-components [CDAT, VisTrails, ParaView, Visit, and R].

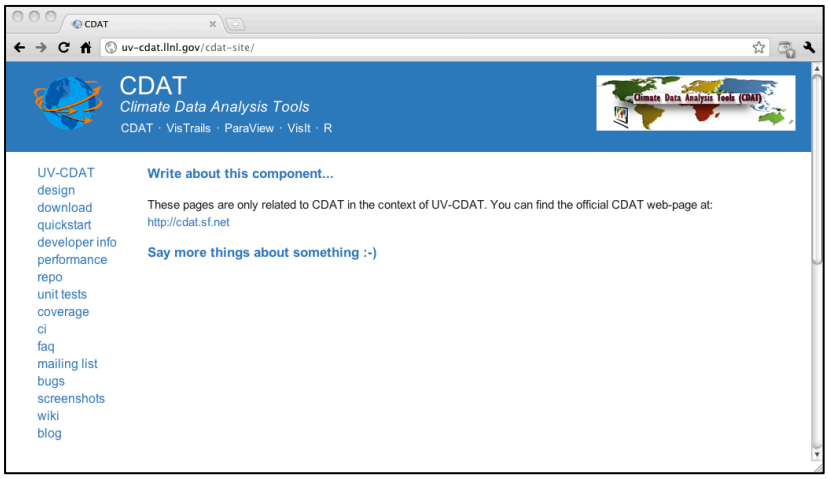


Figure 7: *UV-CDAT component website*

The Wiki is up and running. Some "cleanup" work on the wiki templates is needed to make it more similar to the main UV-CDAT website.

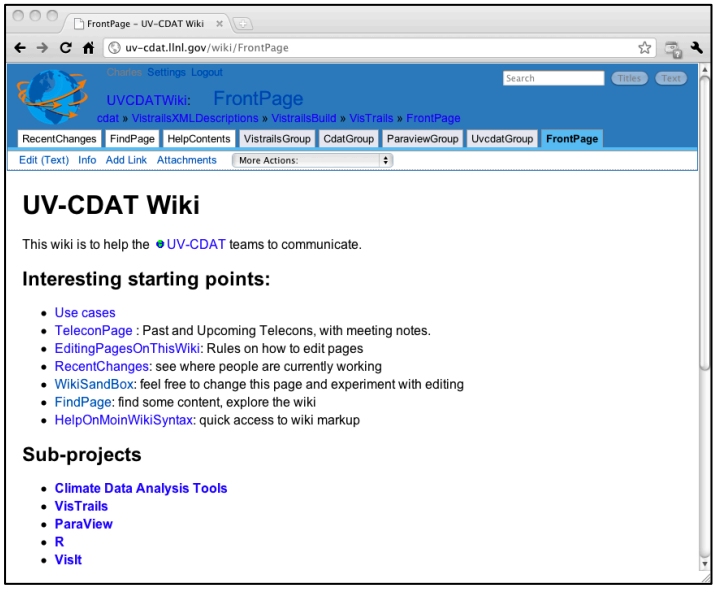


Figure 8: *UV-CDAT's wiki section*

3.10 Code repositories: Git

It has also been agreed that each core component's code will be open-source and distributed via the "git" repository system. The "git" repository URL command is:

- git checkout git@uv-cdat.llnl.gov:uv-cdat.git

Each sub-component needs to be distributed via a git repository as well. At the moment, only CDAT and VisTrails satisfy this requirement. LLNL can easily setup a code repository hosted at LLNL for each UV-CDAT software component.

Once all the sub-component software is in place, the top repository needs to set up a build script that will build all software. Although this is not required, we are pushing for each sub-project to have a "CMake" build mechanism as described above in the "Software Build and Testing" section.

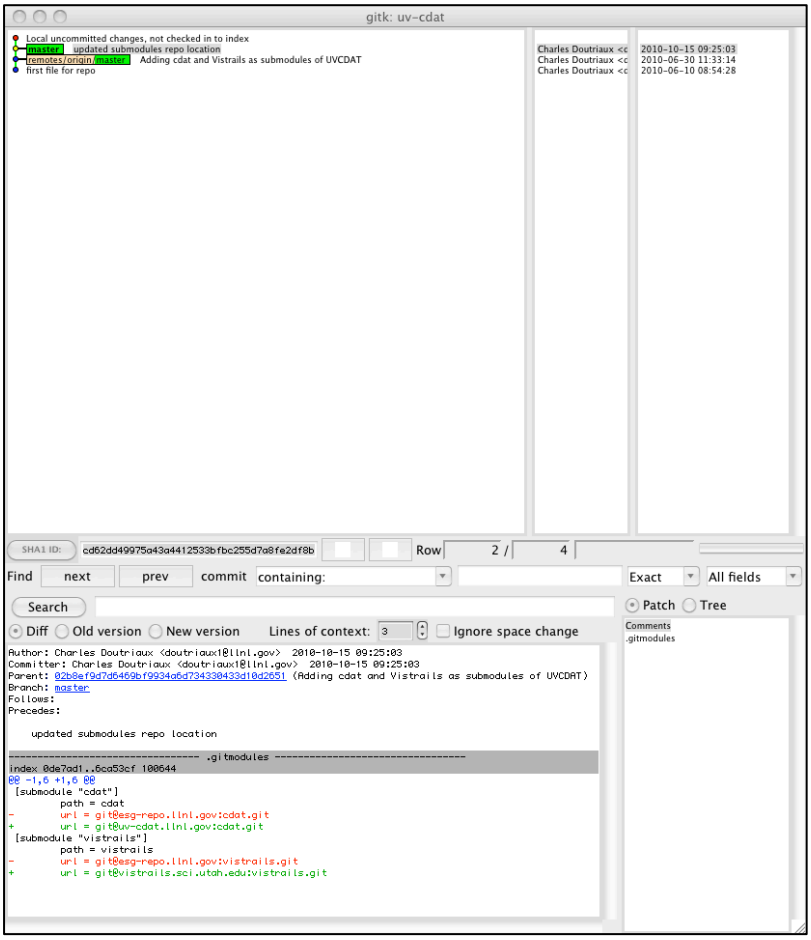


Figure 9: *UV-CDAT git repository*

3.11 Bugs

The "Bugzilla" section of the website has also been set up and operational. Bugzilla will eventually allow users to report bugs and developers to easily identify them and assign their resolution appropriately. We need to connect sendmail to LLNL's mailer so that developers can receive their password for logging onto the system. Sub-projects must

create accounts on Bugzilla, and a "main bug" person must be designated for each component.

Figure 10: *UV-CDAT Bugzilla website*

4 UV-CDAT Group Meetings

The UV-CDAT executive committee holds occasional conference calls each month. These meetings discuss priorities and issues that make up the agendas for project meetings held via teleconference every other Tuesday at 10:00 a.m. Pacific Standard Time. At the project meetings, the entire team discusses project goals, design and development issues, technology, timelines, and milestones. Given the need for more in-depth conversations and examination of work requirements, the following face-to-face meetings were held during this reporting period:

- Kickoff UV-CDAT all-hands meeting: In September, the UV-CDAT project held a three-day, all-hands meeting at Lawrence Berkeley National Laboratory in Berkeley, CA that was dedicated to building collaboration subgroups, user requirements, and project management. The meeting also covered architecture design and set project timelines and milestones.

In addition to our standing project meeting, other subgroups meet regularly. For example, the Parallel I/O team (ORNL, LANL, Kitware) has established weekly meetings to discuss parallel I/O performance enhancements and other general enhancements to the ParaView/VTK systems. These meetings are held each Monday at 4 p.m. Eastern Standard Time.

Besides the UV-CDAT website, mentioned above, a mailing list has also been set up (uv-cdat@lists.llnl.gov), broadcasting announcements and meetings relevant to the entire project.

5 Collaborations

To effectively build an infrastructure that can accommodate ultra-scale visualization and analysis, we established connections with other BER-funded projects and programs at various meetings and workshops, such as the SciDAC 2010 Conference held in Chattanooga, Tennessee, and the more recent SuperComputing 2010 Conference held in New Orleans, Louisiana. More specifically, UV-CDAT coordination and outreach consists of:

- Managing the international execution of the UV-CDAT activity
- Building consensus among modeling groups and arbitrating among various interested groups to maximize scientific value of analysis and visualization
- Providing support services for BER needs and its research and facilitating communication with the modeling groups
- Communicating, interacting with, and responding to scientific groups, government agencies, and a broad range of user groups seeking information about BER analysis and visualization

6 Outreach, Papers, Presentations, and Posters

This section describes the outreach activities for this reporting period in the form of papers published and talks and posters presented.

6.1 Papers

We anticipate reporting papers to national and international peer-review journals in our next progress report.

6.2 Talks

Dean Williams presented UV-CDAT at the Pacific Northwest National Laboratory and LLNL Meeting held in December 2010 at Richland, WA.

Dean Williams presented UV-CDAT at the NASA Technical Data Systems Workshop held in November 2010 at the Goddard Space Flight Center in the Washington, D.C. area.

Dean Williams presented UV-CDAT at the Greenhouse Gas Information System (GHGIS) Mission Operations Center (CMOC) meeting held in October 2010 at Livermore, CA.

6.3 Posters

6.3.1 LLNL Annual Poster Session

Allen Kou, Charles Doutriaux, and Dean N. Williams presented the poster, “Qt Development for Ultra-scale Visualization Climate Data Analysis Tools” at the annual LLNL Research Poster Session, held in Livermore, California in August 2010.